ME 355 – System Modeling and Numerical Methods
Fall 2023

Project 4

Due Friday, December 1 by 11:59 pm
Preliminary functions are due Friday, November 17 by 11:59 pm
This project must be turned in on Blackboard.

1. Introduction

You are working as a freelance contractor specializing in heat transfer problems. You are contracted by a company to develop a piece of Python code that will calculate the rate of cooling of a body over time. You pull open your old heat transfer book and recall the equation

$$\frac{dT}{dt} = -k(T - T_a) \tag{1}$$

where T is the temperature of the body in Celsius (C), t is the time in minutes (min), k is the proportionality constant in min^{-1}, and T_a is the temperature of the surrounding medium in C.

You are told by your contact at the company that the temperature of the surrounding medium can be assumed to fluctuate around room temperature (20 C) by ± 5 C with a period of 2 min and an initial T_a of 25 C at t = 0 min; you decide to model this as a cosine or sine wave. You are also told the proportionality constant for this problem is 0.25 min^{-1}.

Your client is very hesitant about using a black box ordinary differential equation (ODE) solver in Python. Thus, they also request that you create your own ODE solver. You are happy to take up this request, because as a contractor, you are paid by the hour, which means more money!

Your job is to create an ODE solver for a generic Runge-Kutta method. You then need to apply this method to the problem above from $t = 0$ to 20 min. The initial temperature at $t = 0$ min is 90 C. You then wish to compare your method to the ODE solvers in Python.

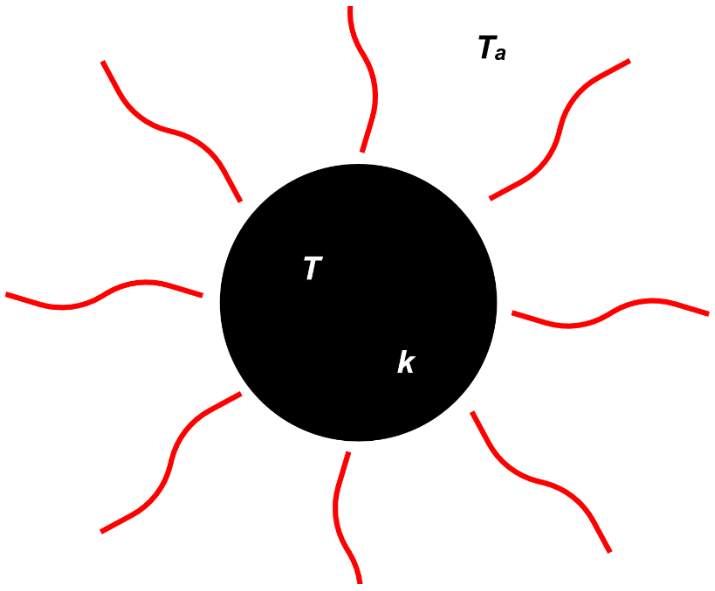

Figure 1. Engineering model of the heat transfer system. The red lines represent the heat transfer between the body and the environment.

2. Functions

To complete this task, you will need to develop **5** Python functions. Preliminary versions of these functions are due on the date specified above. To get full credit for the preliminary functions you must have made a legitimate attempt. You may make changes to the functions after the preliminary function due date and prior to the final due date.

1. **calcK.py**
   K = calcK(f, x, y, h, P, Q)
   This function takes as inputs f which is a function of x, y such that f(x,y) yields a result, x and y which are the xi and yi values respectively, h which is the step size, P which is a 1D list of values such that P = [P1, P2, P3, …..], and Q which is a list of lists such that Q = [[q11], [q21, q22], [q31, q32, q33], …].
   The output of this function is a list of k coefficients such that K = [k1, k2, k3, k4, …]
   If P and Q[-1] are of unequal lengths, ***raise an exception***.

2. **slope.py**
   phi = slope(A, K)
   This function takes inputs of A which is a 1D list of a coefficients such that A = [a1, a2, a3, …] and K which is a 1D list of k coefficients such that K = [k1, k2, k3, k4, …].
   The output is the slope phi used in the generalized Runge-Kutta method such that $y_{i+1} = y_i + \phi h$.

3. **RungeKutta.py**
   [x, y] = RungeKutta(f, x0, y0, h, A, P, Q, n)
   This function performs the generalized Runge-Kutta method such that $y_{i+1} = y_i + \phi h$ for n number of iterations at step size h. The inputs are f which is a function of x,y such that $f(x, y)$ yields a result, x0 and y0 which are the initial x and y values for i = 0, h which is the step size, A which is a 1D list of a coefficients such that A = [a1, a2, …], P which is a 1D list of p coefficients such that P = [p1, p2, p3, …], Q which is a list of lists such that Q = [[q11], [q21, q22], [q31, q32, q33], …], and n which is the number of iterations to perform. The outputs are an array of all x and y values such that x = [x0, x1, …, xn] and y = [y0, y1, …, yn].

4. **Heun.py**
   [A, P, Q] = Heun()
   This function takes no inputs and returns the a, p, and q coefficients for Huen's method such that A = [a1, a2, …], P which is a list of p coefficients such that P = [p1, p2, p3, …], Q which is a list of lists such that Q = [[q11], [q21, q22], [q31, q32, q33], …].

5. **Order3.py**
   [A, P, Q] = Order3()
   This function takes no inputs and returns the a, p, and q coefficients for a 3rd order Runge-Kutta method such that A = [a1, a2, …], P which is a list of p coefficients such that P = [p1, p2, p3, …], Q which is a list of lists such that Q = [[q11], [q21, q22], [q31, q32, q33], …].

3. Figure and Table Documentation

For your project, you want to provide figures and tables with descriptive captions for later use in a report. To complete your documentation, your document should include the following.

1. Title Page
2. Figures
   Each figure should have a **descriptive** caption
   1. Figure 1. This figure should be a T versus t plot of the solution to equation 1 using the values in section 1. You must use your RungeKutta.py file with **Heun's method** using step sizes of 10, 5, 2, 1, and 0.5 min.
      In your caption, comment on the effects of changing step size.
   2. Figure 2. This figure should be a T versus t plot of the solution to equation 1 using the values in section 1. You must plot the solution to your RungeKutta.py file with the **Order 3 method** using a step size of 1 min, Scipy's RK45 method, Scipy's RK23 method, and Scipy's LSODA method. You should use scipy.integrate.solve_ivp() to use the Scipy methods.
      In your caption, briefly compare the 4 methods.
3. Table
   The following table should have a descriptive caption.
   1. Table 1. This table must include information for the 5 different step sizes using the RungeKutta.py file with Heun's method from Figure 1, the 1 step size using the RungeKutta.py file with the Order 3 method from Figure 2, and the 3 Scipy methods from Figure 2. There should be a total of 10 rows including the header. It should include the following columns:
      a. Method
      b. Temperature at 20 min
      c. Number of iterations
      d. Average step size
      In your caption, comment on the accuracy of the different methods as well as the number of iterations and speed of the different methods compared to each other.

4. Deliverables

Be sure to submit all 7 files!

1. calcK.py
2. slope.py
3. RungeKutta.py
4. Heun.py
5. Order3.py
6. Project4.py
7. Figure and Table Documentation PDF

5. Rubric (Total 100 Points)

Function 1 – **15 Points**
Function 2 – **10 Points**
Function 3 – **15 Points**
Function 4 – **5 Points**
Function 5 – **5 Points**
Turned in preliminary functions on time **– 10 points**

Figure and Table Documentation

- ODE problem setup correctly in Project4.py file **– 5 Points**
- Title Page – **5 Points**
- Figure 1 – **10 Points**
- Figure 2 – **10 Points**
- Table 1 – **10 Points**

6. Notes for Document

For any equations, please use the equation editor.
For Figures and Tables, please follow the guidelines outlined in the example report on Blackboard.
Include a title page.

7. Notes for Functions

These notes do not apply to the **project4.py** file.
All functions must be named EXACTLY as listed above. This includes capitalization.
Functional inputs and outputs must be EXACTLY as defined in the Functions section.

8. Test Data

If you run the following code below, you should get the following output. Do note, this does NOT guarantee your code is correct.

```
def h(x,y): return -7*x*y + x
A, P, Q = Heun()
print( RungeKutta(h, 0, 1, 0.5, A, P, Q, 2) ) # ([0, 0.5, 1.0], [1,
0.25, 0.296875])
K = calcK(h,0,1,0.5,P,Q)
print( K ) # [0, -3.0]
print( slope(A,K) ) # -1.5
```